Restrictions on Tree Adjoining Languages

Giorgio Satta
Dip. di Elettronica e Informatica
Università di Padova
35131 Padova, Italy
satta@dei.unipd.it

William Schuler
Computer and Information Science Dept.
University of Pennsylvania
Philadelphia, PA 19103
schuler@linc.cis.upenn.edu

Abstract

Several methods are known for parsing languages generated by Tree Adjoining Grammars (TAGs) in $\mathcal{O}(n^6)$ worst case running time. In this paper we investigate which restrictions on TAGs and TAG derivations are needed in order to lower this $\mathcal{O}(n^6)$ time complexity, without introducing large runtime constants, and without losing any of the generative power needed to capture the syntactic constructions in natural language that can be handled by unrestricted TAGs. In particular, we describe an algorithm for parsing a strict subclass of TAG in $\mathcal{O}(n^5)$, and attempt to show that this subclass retains enough generative power to make it useful in the general case.

1 Introduction

Several methods are known that can parse languages generated by Tree Adjoining Grammars (TAGs) in worst case time $\mathcal{O}(n^6)$, where n is the length of the input string (see (Schabes and Joshi, 1991) and references therein). Although asymptotically faster methods can be constructed, as discussed in (Rajasekaran and Yooseph, 1995), these methods are not of practical interest, due to large hidden constants. More generally, in (Satta, 1994) it has been argued that methods for TAG parsing running in time asymptotically faster than $\mathcal{O}(n^6)$ are unlikely to have small hidden constants.

A careful inspection of the proof provided in (Satta, 1994) reveals that the source of the claimed computational complexity of TAG parsing resides in the fact that auxiliary trees can get adjunctions at (at least) two distinct nodes in their spine (the path connecting the root and the foot nodes). The question then arises of whether the bound of two is tight. More generally, in this paper we investigate which restrictions on TAGs are needed in order to lower the $\mathcal{O}(n^6)$ time complexity, still retaining the generative power that is needed to capture the syntactic constructions of natural language that unrestricted TAGs can handle. The contribution of this paper is twofold:

- We define a strict subclass of TAG where adjunction of so-called wrapping trees at the spine is restricted to take place at no more than one distinct node. We show that in this case the parsing problem for TAG can be solved in worst case time $\mathcal{O}(n^5)$.
- We provide evidence that the proposed subclass still captures the vast majority of TAG analyses that have been currently proposed for the syntax of English and of several other languages.

Several restrictions on the adjunction operation for TAG have been proposed in the literature (Schabes and Waters, 1993; Schabes and Waters, 1995) (Rogers, 1994). Differently from here, in all those works the main goal was one of characterizing, through the adjunction operation, the set of trees that can be generated by a context-free grammar (CFG). For the sake of critical comparison, we discuss some common syntactic constructions found in current natural language TAG analyses, that can be captured by our proposal but fall outside of the restrictions mentioned above.

2 Overview

We introduce here the subclass of TAG that we investigate in this paper, and briefly compare it with other proposals in the literature.

A TAG is a tuple $G = (N, \Sigma, I, A, S)$, where N, Σ are the finite sets of nonterminal and terminal symbols, respectively, I, A are the finite

sets of initial and auxiliary trees, respectively, and $S \in N$ is the initial symbol. Trees in $I \cup A$ are also called elementary trees. The reader is referred to (Joshi, 1985) for the definitions of tree adjunction, tree substitution, and language derived by a TAG.

The **spine** of an auxiliary tree is the (unique) path that connects the root and the foot node. An auxiliary tree β is called a **right** (**left**) tree if (i) the leftmost (rightmost, resp.) leaf in β is the foot node; and (ii) the spine of β contains only the root and the foot nodes. An auxiliary tree which is neither left nor right is called a **wrapping** tree.[1]

The **TAG restriction** we propose is stated as followed:

1. At the spine of each wrapping tree, there is at most one node that can host adjunction of a wrapping tree. This node is called a **wrapping** node.

2. At the spine of each left (right) tree, no wrapping tree can be adjoined and no adjunction constraints on right (left, resp.) auxiliary trees are found.

The above restriction does not in any way constrain adjunction at nodes that are not in the spine of an auxiliary tree. Similarly, there is no restriction on the adjunction of left or right trees at the spines of wrapping trees.

Our restriction is fundamentally different from those in (Schabes and Waters, 1993; Schabes and Waters, 1995) and (Rogers, 1994), in that we allow wrapping auxiliary trees to nest inside each other an unbounded number of times, so long as they only adjoin at one place in each others' spines. Rogers, in contrast, restricts the nesting of wrapping auxiliaries to a number of times bounded by the size of the grammar, and Schabes and Waters forbid wrapping auxiliaries altogether, at any node in the grammar.

We now focus on the recognition problem, and informally discuss the computational advantages that arise in this task when a TAG obeys the above restriction. These ideas are formally developed in the next section. Most of the tabular methods for TAG recognition represent subtrees of derived trees, rooted at some node N and having the same span within the input string, by means of items of the form $\langle N, i, p, q, j \rangle$. In this notation i, j are positions in the input spanned by N, and p, q are positions spanned by the foot node, in case N belongs to the spine, as we assume in the discussion below.

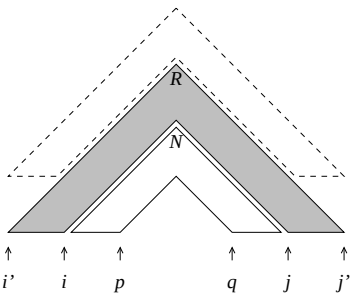

Figure 1: $\mathcal{O}(n^6)$ wrapping adjunction step.

The most time expensive step in TAG recognition is the one that deals with adjunction. When we adjoin at N a derived auxiliary tree rooted at some node R, we have to combine together two items $\langle R, i', i, j, j' \rangle$ and $\langle N, i, p, q, j \rangle$. This is shown in Figure 1. This step involves six different indices that could range over any position in the input, and thus has a time cost of $\mathcal{O}(n^6)$.

Let us now consider adjunction of wrapping trees, and leave aside left and right trees for the moment. Assume that no adjunction has been performed in the portion of the spine below N. Then none of the trees adjoined below N will simultaneously affect the portions of the tree yield to the left and to the right of the foot node. In this case we can safely split the tree yield and represent item $\langle N, i, p, q, j \rangle$ by means of two items of a new kind, $\langle N_{left}, i, p \rangle$ and $\langle N_{right}, q, j \rangle$. The adjunction step can now be performed by means of two successive steps. The first step combines $\langle R, i', i, j, j' \rangle$ and $\langle N_{left}, i, p \rangle$, producing a new intermediate item I. The second step combines I and $\langle N_{right}, q, j \rangle$, producing the desired result. In this way the time cost is reduced to $\mathcal{O}(n^5)$.

It is not difficult to see that the above reasoning also applies in cases where no adjunction has been performed at the portion of the spine above N. This suggests that, when pro-

[1]The above names are also used in (Schabes and Waters, 1995) for slightly different kinds of trees.

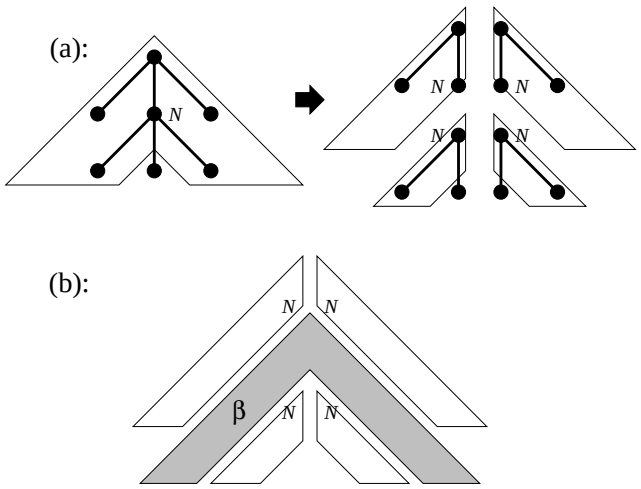

Figure 2: $\mathcal{O}(n^5)$ wrapping adjunction step.

cessing a TAG that obeys the restriction introduced above, we can always 'split' each wrapping tree into four parts at the wrapping node N, since N is the only site in the spine that can host adjunction (see Figure 2(a)). Adjunction of a wrapping tree β at N can then be simulated by four steps, executed one after the other. Each step composes the item resulting from the application of the previous step with an item representing one of the four parts of the wrapping tree (see Figure 2(b)).

We now consider adjunction involving left and right trees, and show that a similar splitting along the spine can be performed. Assume that γ is a derived auxiliary tree, obtained by adjoining several left and right trees one at the spine of the other. Let x and y be the part of the yield of γ to the left and right, respectively, of the foot node. From the definition of left and right trees, we have that the nodes in the spine of γ have all the same nonterminal label. Also, from condition 2 in the above restriction we have that the left trees adjoined in γ do not constrain in any way the right trees adjoined in γ. Then the following derivation can always be performed. We adjoin all the left trees, each one at the spine of the other, in such a way that the resulting tree γ_{left} has yield x. Similarly, we adjoining all the right trees, one at the spine of the other, in such a way that the yield of the resulting tree γ_{right} is y. Finally, we adjoin γ_{right} at the root of γ_{left}, obtaining a derived tree having the same yield as γ.

From the above observations it directly follows that we can always recognize the yield of γ by independently recognizing γ_{left} and γ_{right}. Most important, γ_{left} and γ_{right} can be represented by means of items $\langle R_{left}, i, p\rangle$ and $\langle R_{right}, q, j\rangle$. As before, the adjunction of tree γ at some subtree represented by an item I can be recognized by means of two successive steps, one combining I with $\langle R_{left}, i, p\rangle$ at its left, resulting in an intermediate item I', and the second combining I' with $\langle R_{right}, q, j\rangle$ at its right, obtaining the desired result.

3 Recognition

This section presents the main result of the paper. We provide an algorithm for the recognition of languages generated by the subclass of TAGs introduced in the previous section, and show that the worst case running time is $\mathcal{O}(n^5)$, where n is the length of the input string. To simplify the presentation, we assume the following conditions throughout this section: first, that elementary trees are binary (no more than two children at each node) and no leaf node is labeled by ε; and second, that there is always a wrapping node in each wrapping tree, and it differs from the foot and the root node. This is without any loss of generality.

3.1 Grammar transformation

Let $G = (N, \Sigma, I, A)$ be a TAG obeying the restrictions of Section 2. We first transform A into a new set of auxiliary trees A' that will be processed by our method. The root and foot nodes of a tree β are denoted R_β and F_β, respectively. The wrapping node (as defined in Section 2) of β is denoted W_β.

Each left (right) tree β in A is inserted in A' and is called β_L (β_R). Let β be a wrapping tree in A. We split β into four auxiliary trees, as informally described in Section 2. Let β_D be the subtree of β rooted at W_β. We call β_U the tree obtained from β by removing every descendant of W_β (and the corresponding arcs). We remove every node to the right (left) of the spine of β_D and call β_{LD} (β_{RD}) the resulting tree. Similarly, we remove every node to the right (left) of the spine of β_U and call β_{LU} (β_{RU}) the resulting tree. We set $F_{\beta_{LD}}$ and $F_{\beta_{RD}}$ equal to F_β, and set $F_{\beta_{LU}}$ and $F_{\beta_{RU}}$ equal to W_β. Trees β_{LU}, β_{RU}, β_{LD}, and β_{RD} are inserted in A' for every wrapping tree β in A.

Each tree in A' inherits at its nodes the adjunction constraints specified in G. In addition, we impose the following constraints:

- only trees β_L can be adjoined at the spine of trees β_{LD}, β_{LU};
- only trees β_R can be adjoined at the spine of trees β_{RD}, β_{RU};
- no adjunction can be performed at nodes $F_{\beta_{LU}}, F_{\beta_{RU}}$.

3.2 The algorithm

The algorithm below is a tabular method that works bottom up on derivation trees. Following (Shieber et al., 1995), we specify the algorithm using inference rules. (The specification has been optimized for presentation simplicity, not for computational efficiency.)

Symbols N, P, Q denote nodes of trees in A' (including foot and root), α denotes initial trees and β denotes auxiliary trees. Symbol $label(N)$ is the label of N and $children(N)$ is a string denoting all children of N from left to right ($children(N)$ is undefined if N is a leaf). We write $\alpha \in Sbst(N)$ if α can be substituted at N. We write $\beta \in Adj(N)$ if β can be adjoined at N, and nil $\in Adj(N)$ if adjunction at N is optional.

We use two kind of items:

- Item $\langle N^X, i, j\rangle$, $X \in \{B, M, T\}$, denotes a subtree rooted at N and spanning the portion of the input from i to j. Note that two input positions are sufficient, since trees in A' always have their foot node at the position of the leftmost or rightmost leaf. We have $X = B$ if N has not yet been processed for adjunction, $X = M$ if N has been processed only for adjunction of trees β_L, and $X = T$ if N has already been processed for adjunction.
- Item $\langle \beta, i, p, q, j\rangle$ denotes a wrapping tree β (in A) with R_β spanning the portion of the input from i to j and with F_β spanning the portion of the input from p to q. In place of β we might use symbols $[\beta, LD]$, $[\beta, RD]$ and $[\beta, RU]$ to denote the temporary results of recognizing the adjunction of some wrapping tree at W_β.

Algorithm. Let G be a TAG with the restrictions of Section 2, and let A' be the associated set of auxiliary trees defined as in section 3.1. Let $a_1 a_2 \cdots a_n$, $n \geq 1$, be an input string. The algorithm accepts the input iff some item $\langle R^T_\alpha, 0, n\rangle$ can be inferred for some $\alpha \in I$.

Step 1 This step recognizes subtrees with root N from subtrees with roots in $children(N)$.

$$\frac{}{\langle N^T, i-1, i\rangle},\ label(N) = a_i;$$

$$\frac{}{\langle F^B_\beta, i, i\rangle},\ \beta \in A',\ 0 \leq i \leq n;$$

$$\frac{\langle R^T_\alpha, i, j\rangle}{\langle N^T, i, j\rangle},\ \alpha \in Sbst(N);$$

$$\frac{\langle P^T, i, k\rangle \quad \langle Q^T, k, j\rangle}{\langle N^B, i, j\rangle},\ children(N) = PQ;$$

$$\frac{\langle P^T, i, j\rangle}{\langle N^B, i, j\rangle},\ children(N) = P.$$

Step 2 This step recognizes the adjunction of wrapping trees at wrapping nodes. We recognize the tree hosting adjunction by composing its four 'chunks', represented by auxiliary trees β_{LD}, β_{RD}, β_{RU} and β_{LU} in A', around the wrapped tree.

$$\frac{\langle R^B_{\beta_{LD}}, k, p\rangle \quad \langle \beta', i, k, q, j\rangle}{\langle [\beta, LD], i, p, q, j\rangle}, \beta' \in Adj(W_\beta), p < q;$$

$$\frac{\langle R^B_{\beta_{RD}}, q, k\rangle \quad \langle [\beta, LD], i, p, k, j\rangle}{\langle [\beta, RD], i, p, q, j\rangle},\ p < q;$$

$$\frac{\langle R^T_{\beta_{RU}}, k, j\rangle \quad \langle [\beta, RD], i, p, q, k\rangle}{\langle [\beta, RU], i, p, q, j\rangle};$$

$$\frac{\langle R^T_{\beta_{LU}}, i, k\rangle \quad \langle [\beta, RU], k, p, q, j\rangle}{\langle \beta, i, p, q, j\rangle};$$

$$\frac{\langle R^T_{\beta_{LD}}, i, p\rangle \quad \langle R^T_{\beta_{RD}}, q, j\rangle}{\langle [\beta, RD], i, p, q, j\rangle}, \text{nil} \in Adj(W_\beta), p < q.$$

Step 3 This step recognizes all remaining cases of adjunction.

$$\frac{\langle R^T_{\beta_L}, i, k\rangle \quad \langle N^B, k, j\rangle}{\langle N^X, i, j\rangle}, \beta \in Adj(N), X \in \{M, T\};$$

$$\frac{\langle N^X, i, k\rangle \quad \langle R^T_{\beta_R}, k, j\rangle}{\langle N^T, i, j\rangle}, \beta \in Adj(N), X \in \{B, M\};$$

$$\frac{\langle N^B, i, j\rangle}{\langle N^T, i, j\rangle},\ \text{nil} \in Adj(N);$$

$$\frac{\langle N^B, p, q\rangle \quad \langle \beta, i, p, q, j\rangle}{\langle N^T, i, j\rangle},\ \beta \in Adj(N).$$

Due to restrictions on space, we merely claim the correctness of the above algorithm. We now establish its worst case time complexity with respect to the input string length n. We need to consider the maximum number d of input positions appearing in the antecedent of an inference rule. In fact, in the worst case we will have to execute a number of different evaluations of each

inference rule which is proportional to n^d, and each evaluation can be carried out in an amount of time independent of n. It is easy to establish that Step 1 can be executed in time $\mathcal{O}(n^3)$ and that Step 3 can be executed in time $\mathcal{O}(n^4)$. Adjunction at wrapping nodes performed at Step 2 is the most expensive operation, requiring an amount of time $\mathcal{O}(n^5)$. This is also the time complexity of our algorithm.

4 Linguistic Relevance

In this section we will attempt to show that the restricted formalism presented in Section 2 retains enough generative power to make it useful in the general case.

4.1 Athematic and Complement Trees

We begin by introducing the distinction between athematic auxiliary trees and complement auxiliary trees (Kroch, 1989), which are meant to exhaustively characterize the auxiliary trees used in any natural language TAG grammar.[2] An **athematic** auxiliary tree does not subcategorize for or assign a thematic role to its foot node, so the head of the foot node becomes the head of the phrase at the root. The structure of an athematic auxiliary tree may thus be described as:

$$X^n \rightarrow X^n \cdots (Y^{max}) \cdots, \qquad (1)$$

where X^n is any projection of category X, Y^{max} is the maximal projection of Y, and the order of the constituents is variable.[3] A **complement** auxiliary tree, on the other hand, introduces a lexical head that subcategorizes for the tree's foot node and assigns it a thematic role. The structure of a complement auxiliary tree may be described as:

$$X^{max} \rightarrow \cdots Y^0 \cdots X^{max} \cdots, \qquad (2)$$

where X^{max} is the maximal projection of some category X, and Y^0 is the lexical projection of some category Y, whose maximal projection dominates X^{max}.

From this we make the following observations:

1. Because it does not assign a theta role to its foot node, an athematic auxiliary tree may adjoin at any projection of a category, which we take to designate any adjunction site in a host elementary tree.
2. Because it does assign a theta role to its foot node, a complement auxiliary tree may only adjoin at a certain 'complement' adjunction site in a host elementary tree, which must at least be a maximal projection of a lexical category.
3. The foot node of an athematic auxiliary tree is dominated only by the root, with no intervening nodes, so it falls outside of the maximal projection of the head.
4. The foot node of a complement auxiliary tree is dominated by the maximal projection of the head, which may also dominate other arguments on either side of the foot.

To this we now add the assumption that each auxiliary tree can have only one complement adjunction site projecting from Y^0, where Y^0 is the lexical category that projects Y^{max}. This is justified in order to prevent projections of Y^0 from receiving more than one theta role from complement adjuncts, which would violate the underlying theta criterion in Government and Binding Theory (Chomsky, 1981).We also assume that an auxiliary tree can not have complement adjunction sites on its spine projecting from lexical heads other than Y^0, in order to preserve the minimality of elementary trees (Kroch, 1989; Frank, 1992). Thus there can be no more than one complement adjunction site on the spine of any complement auxiliary tree, and no complement adjunction site on the spine of any athematic auxiliary tree, since the foot node of an athematic tree lies outside of the maximal projection of the head.[4]

[2] The same linguistic distinction is used in the conception of 'modifier' and 'predicative' trees (Schabes and Shieber, 1994), but Schabes and Shieber give the trees special properties in the calculation of derivation structures, which we do not.

[3] The CFG-like notation is taken directly from (Kroch, 1989), where it is used to specify labels at the root and frontier nodes of a tree without placing constraints on the internal structure.

[4] It is important to note that, in order to satisfy the theta criterion and minimality, we need only constrain the number of complement adjunctions – not the number of complement adjunction sites – on the spine of an auxiliary tree. Although this would remain within the power of our formalism, we prefer to use constraints expressed in terms of adjunction sites, as we did in Section 2, be-

Based on observations 3 and 4, we can further specify that only complement trees may wrap, because the foot node of an athematic tree lies outside of the maximal projection of the head, below which all of its subcategories must attach.[5] In this manner, we can insure that only one wrapping tree (the complement auxiliary) can adjoin into the spine of a wrapping (complement) auxiliary, and only athematic auxiliaries (which must be left/right trees) can adjoin elsewhere, fulfilling our TAG restriction in Section 2.

4.2 Possible Extensions

We may want to weaken our definition to include wrapping athematic auxiliaries, in order to account for modifiers with raised heads or complements as in Figure 3: "They so revered him that they built a statue in his honor." This can be done within the above algorithm as long as the athematic trees do not wrap productively (that is as long as they cannot be adjoined one at the spine of the other) by splitting the athematic auxiliary tree down the spine and treating the two fragments as tree-local multicomponents, which can be simulated with non-recursive features (Hockey and Srinivas, 1993).

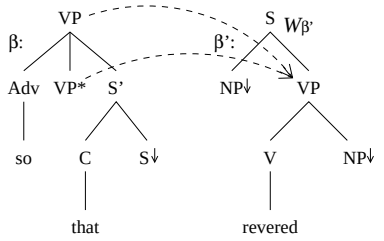

Figure 3: Wrapping athematic tree.

Since the added features are non-recursive, this extension would not alter the $\mathcal{O}(n^5)$ result reported in Section 3.

4.3 Comparison of Coverage

In contrast to the formalisms of Schabes and Waters (Schabes and Waters, 1993; Schabes and Waters, 1995), our restriction allows wrapping complement auxiliaries as in Figure 4 (Schabes and Waters, 1995). Although it is difficult to find examples in English which are excluded by Rogers' regular form restriction (Rogers, 1994), we can cite verb-raised complement auxiliary trees in Dutch as in Figure 5 (Kroch and Santorini, 1991). Trees with this structure may adjoin into each others' internal spine nodes an unbounded number of times, in violation of Rogers' definition of regular form adjunction, but within our criteria of wrapping adjunction at only one node on the spine.

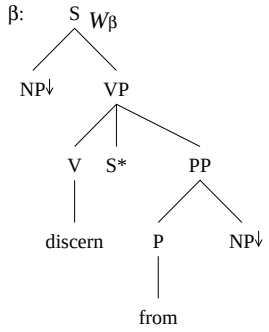

Figure 4: Wrapping complement tree.

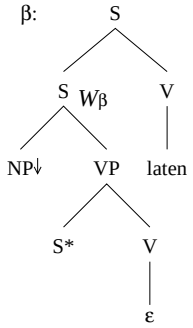

Figure 5: Verb-raising tree in Dutch.

5 Concluding remarks

Our proposal is intended to contribute to the assessment of the computational complexity of syntactic processing. We have introduced a strict subclass of TAGs having the generative power that is needed to account for the syntactic constructions of natural language that unrestricted TAGs can handle. We have specified a method that recognizes the generated languages in worst case time $\mathcal{O}(n^5)$, where n is the length of the input string. In order to account for the dependency on the input grammar G, let us define $|G| = \sum_N (1 + |\,Adj(N)\,|)$, where N ranges over the set of all nodes of the elementary trees.

cause it provides a restriction on elementary trees, rather than on derivations.

[5] Except in the case of raising, discussed below.

It is not difficult to see that the running time of our method is proportional to $|G|$.

Our method works as a recognizer. As for many other tabular methods for TAG recognition, we can devise simple procedures in order to obtain a derived tree associated with an accepted string. To this end, we must be able to 'interleave' adjunctions of left and right trees, that are always kept separate by our recognizer.

The average case time complexity of our method should surpass its worst case time performance, as is the case for many other tabular algorithms for TAG recognition. In a more applicative perspective, then, the question arises of whether there is any gain in using an algorithm that is unable to recognize more than one wrapping adjunction at each spine, as opposed to using an unrestricted TAG algorithm. As we have tried to argue in Section 4, it seems that standard syntactic constructions do not exploit multiple wrapping adjunctions at a single spine. Nevertheless, the local ambiguity of natural language, as well as cases of ill-formed input, could always produce cases in which such expensive analyses are attempted by an unrestricted algorithm. In this perspective, then, we conjecture that having the single-wrapping-adjunction restriction embedded into the recognizer would improve processing efficiency in the average case. Of course, more experimental work would be needed in order to evaluate such a conjecture, which we leave for future work.

Acknowledgments

Part of this research was done while the first author was visiting the Institute for Research in Cognitive Science, University of Pennsylvania. The first author was supported by NSF grant SBR8920230. The second author was supported by U.S. Army Research Office Contract No. DAAH04-94G-0426. The authors would like to thank Christy Doran, Aravind Joshi, Anthony Kroch, Mark-Jan Nederhof, Marta Palmer, James Rogers and Anoop Sarkar for their help in this research.

References

Noam Chomsky. 1981. *Lectures on government and binding*. Foris, Dordercht.

Robert Frank. 1992. *Syntactic locality and tree adjoining grammar: grammatical acquisition and processing perspectives*. Ph.D. thesis, Computer Science Department, University of Pennsylvania.

Beth Ann Hockey and Srinivas Bangalore. 1993. Feature-based TAG in place of multi-component adjunction: computational implications. In *Proceedings of the Natural Language Processing Pacific Rim Symposium (NLPRS)*, Fukuoka, Japan.

Aravind K. Joshi. 1985. How much context sensitivity is necessary for characterizing structural descriptions: Tree adjoining grammars. In L. Karttunen D. Dowty and A. Zwicky, editors, *Natural language parsing: Psychological, computational and theoretical perspectives*, pages 206–250. Cambridge University Press, Cambridge, U.K.

Anthony S. Kroch and Beatrice Santorini. 1991. The derived constituent structure of west germanic verb-raising construction. In Robert Freidin, editor, *Principles and Parameters in Comparative Grammar*, pages 269–338. MIT Press.

Anthony S. Kroch. 1989. Asymmetries in long distance extraction in a TAG grammar. In M. Baltin and A. Kroch, editors, *Alternative Conceptions of Phrase Structure*, pages 66–98. University of Chicago Press.

Sanguthevar Rajasekaran and Shibu Yooseph. 1995. TAL recognition in $\mathcal{O}(M(n^2))$ time. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL '95)*.

James Rogers. 1994. Capturing CFLs with tree adjoining grammars. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL '94)*.

Giorgio Satta. 1994. Tree adjoining grammar parsing and boolean matrix multiplication. *Computational Linguistics*, 20(2):173–192.

Yves Schabes and Aravind K. Joshi. 1991. Parsing with lexicalized tree adjoining grammar. In M. Tomita, editor, *Current Issues in Parsing Technologies*. Kluwer Academic Publishers.

Yves Schabes and Stuart M. Shieber. 1994. An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20(1):91–124.

Yves Schabes and Richard C. Waters. 1993. Lexicalized context-free grammars. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL '93)*.

Yves Schabes and Richard C. Waters. 1995. Tree insertion grammar: A cubic-time parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*, 21(4):479–515.

Stuart M. Shieber, Yves Schabes, and Fernando C.N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24:3–36.